Complexity of Problems for Commutative Grammars

Eryk Kopczyński*
Institute of Informatics, Warsaw University
erykk@mimuw.edu.pl

Abstract

We consider Parikh images of languages accepted by non-deterministic finite automata and context-free grammars; in other words, we treat the languages in a commutative way — we do not care about the order of letters in the accepted word, but rather how many times each one of them appears. In most cases we assume that the alphabet is of fixed size. We show tight complexity bounds for problems like membership, equivalence, and disjointness. In particular, we show polynomial algorithms for membership and disjointness for Parikh images of non-deterministic finite automata over fixed alphabet, and we show that equivalence is Π_2^{P} complete for context-free grammars over fixed terminal alphabet.

1 Introduction

We consider languages accepted by regular and context-free grammars, except that we treat the language in a commutative way — we do not care about the order of letters in the accepted word, but rather how many times each one of them appears. In this setting, usual problems, like membership and equivalence, have different complexities than in the non-commutative case.

A well known classic result in this area is the result of Parikh [Par66] that, for a context-free grammar G over alphabet Σ, the Parikh image of G, i.e., the set $\mathrm{out}(G) \subseteq \mathbb{N}^{\Sigma}$ of such multisets M that, in some word $w \in L(G)$, each letter x appears $M(x)$ times, is a semilinear set. Some complexity results regarding semilinear sets and commutative grammars have been obtained by D. Huynh [Hu80, Hu85], who has shown that equivalence is Π_2^{P}-hard both for semilinear sets and commutative grammars (where Π_2^{P} is the dual of the second level of the polynomial-time hierarchy, [Sto77]).

There are many practical uses of regular and context-free languages which do not care about the order of the letters in the word. For example, when considering regular languages of trees, we might be not interested in the ordering of children of a given node. [BM99] and [NS99] consider XML schemas allowing marking some nodes as unordered.

Some research has also been done in the field of communication-free Petri nets, or Basic Parallel Processes (BPP). A Petri net ([Pet81], [Rei85]) is communication-free if each transition has only one input. This restriction means that such a Petri net is essentially equivalent to a commutative context-free grammar. [Yen96] shows that the reachability equivalence problem for BPP-nets can be solved in $DTIME\left(2^{2^{ds^3}}\right)$. For general Petri nets, reachability (membership in terms of grammars) is decidable [Ko82], although the known algorithms require non-primitive recursive space; and reachability equivalence is undecidable [Ha76]. Also, some harder types of equivalence problems are undecidable for BPP nets [Hü94]. See [EN93] for a survey of decidability results regarding Petri nets.

It turns out that, contrary to the non-commutative case, the size of alphabet is very important. In the non-commutative case, we can use strings a, ab, and abb to encode a three letter alphabet $\{a, b, c\}$ using two letters. Trying to do this in the commutative case fails, since two different words ac and bb are mapped to $aabb$ and $abab$, which are commutatively the same word. There is no way to map a three letter alphabet to a two letter one which does not collapse anything. Each new letter adds a new dimension to the problem in the commutative case — literally: commutative words (multisets) over an alphabet of size d are better viewed as points in a d-dimensional space, rather than strings.

Contrary to most previous papers on commutative grammars, in most cases we assume that our (terminal) alphabet is of fixed size. As far as we know, there have been no successful previous attempts in this direction (except for the much simpler case $d = 1$ [Hu84]). Our methods enable us to obtain tight complexity bounds for most of the basic problems (like membership, inclusion, equivalence, universality, disjointness) for both regular and context-free commutative grammars, over an alphabet of fixed size. In some cases, we provide algorithms for the special case $d = 2$, as

*Supported by the Polish government grant no. N206 008 32/0810

they are much simpler than the general ones.

In Theorem 5.3, we show a polynomial algorithm deciding membership for regular languages, i.e., whether a multiset (given as a binary representation) is in the Parikh image of a regular language (represented by the non-deterministic finite automaton accepting it).

In Theorem 6.2, we improve upon Parikh's result quoted above in two ways, assuming that the alphabet is of size 2. First, $\mathrm{out}(G)$ is produced as a union of linear sets with only two periods (whose magnitude is single exponential in size of G); second, these linear sets can be grouped in a polynomial number of bundles such that each bundle shares the pairs of periods used (we call such a bundle an A, B-frame). This leads to a Π_2^{P}algorithm for solving inclusion (equivalence) for context-free languages. Unfortunately, such simple presentation is impossible for alphabets of size greater than 2; we provide a counterexample where $d = 3$, and a much more complicated reasoning which solves the general case (still resulting in a Π_2^{P}algorithm).

The following table summarizes our results. Alphabet size F means that alphabet is of fixed size, and U means unfixed size. We consider the basic problems: membership, universality, inclusion, and disjointness; note that solving inclusion is equivalent to solving equivalence — simple reductions exist in both ways. We use c as an abbreviation for complete. Our main results — our most important algorithms — are marked with bold (polynomial algorithms for checking membership and disjointness for regular grammars over alphabets of fixed size, Π_2^{P}-completeness of the inclusion (equivalence) problems for context-free grammars over alphabets of fixed size). Problems which have been shown to be hard are marked with stars (NP-completeness of membership checking for regular grammars over alphabets of unfixed size and context-free grammars over alphabets of size 1, coNP-completeness of universality checking for regular grammars over alphabets of size 1, Π_2^{P}-completeness of inclusion (equivalence) checking for context-free grammars); the proofs are simple and have been included for sake of completeness.

regular languages				
alphabet size	1	2	F	U
membership	P	P	**P**	NPc*
universality	coNPc*	coNPc	coNPc	?
inclusion	coNPc	coNPc	coNPc	?
disjointness	P	P	**P**	coNPc
context-free languages				
alphabet size	1	2	F	U
membership	NPc*	NPc	NPc	NPc
universality	Π_2^{P}	Π_2^{P}	Π_2^{P}	?
inclusion	Π_2^{P}c	$\mathbf{\Pi_2^P c}$	$\mathbf{\Pi_2^P c}$	?
disjointness	coNPc	coNPc	coNPc	?

2 Overview

In this section, we present our techniques and results in an informal way. The formal version can be found in the following sections.

Our main observation is that we can treat our runs (or derivations in CF grammars) completely commutatively: we just count how many times each transition (rule) has been used. In both cases, the validity of such a „commutative run" can be checked by checking two very simple conditions: Euler condition (each state is entered as many times as it is used) and connectedness (there are no unconnected loops) — Theorem 4.1. From this, we immediately get that checking membership of a given multiset in a Parikh image of a context-free language is in NP.

The second observation is that we can decompose a run into smaller parts, ultimately obtaining its *skeleton*, to which we add some (simple) *cycles*. Since the skeleton uses all states that the original run used (which we call its *support*), we can add these cycles in arbitrary numbers to our skeleton, always getting valid runs. Moreover, in case of finite automata (regular grammars), both the skeleton and the cycles are bounded polynomially (in the size of the automaton, $|G|$ — Lemma 5.1).

Now, linear algebraic considerations come into play. Whenever we have d linearly independent vectors $v_1, \ldots, v_d$ with integer coordinates in a d-dimensional space, for each other vector v, there is a constant C such that Cv can be written as a linear combination of $v_1, \ldots, v_d$ with integer coefficients (Lemma 3.1). This C is bounded polynomially by coordinates of $v_1, \ldots, v_d$ (d appears in the exponent). In our case, our vectors will be the Parikh images of our cycles, with d letters in our alphabet.

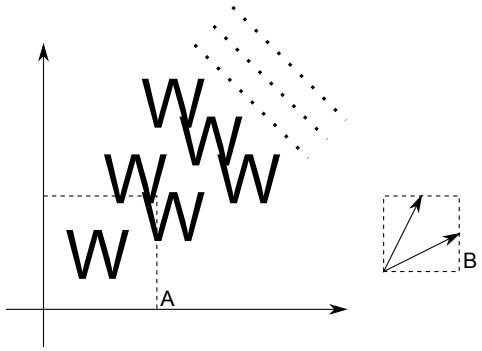
Thus, whenever we have a non-negative integer combination of more than d cycles, where the multiplicities of cycles are big enough, we can reconstruct our Parikh image using different multiplicities of these cycles, and do such „shifting" until the multiplicities of some cycles drop (Lemma 3.2 and Theorem 5.2). Thus, there are at most d cycles which we are using in large quantities. From this, we get an algorithm for membership (Theorem 5.3): we can guess the small run and the d cycles (making sure that the run crosses these cycles), and then just check whether we obtain our result by adding the cycles in non-negative integer amounts to our run — which boils down to solving a system of equations. This algorithm is polynomial, since everything it uses is bounded polynomially.

The situation is the simplest in the case of $d = 2$, where instead of guessing the two cycles, we can always use the two extreme ones v_a, v_b — i.e., the ones which proportionally contain the greatest quantities of the two letters a and b in our alphabet. Each other cycle can be written as a non-negative combination of these two. Still, we have to

take the extreme cycles which cross our small run — which means that we have to guess the two states where they cross, and then take extreme cycles crossing these states. For unfixed d, or for context-free grammars, the problem is NP-complete.

Now, what about context-free grammars? Generally, we can use the same techniques, but now, skeletons and cycles are bounded exponentially. In case of $d = 2$, we get Theorem 6.2: a Parikh image of G is a union of a polynomial number of A, B-*frames*; where an A, B-frame (Definition 3.3) is a set of vectors defined by some W (a subset of $\mathbb{N}^d$ bounded by A) and two vectors v_a, v_b (bounded by B), consisting of all vectors of form $w + n_a v_a + n_b v_b$ (where $w \in W$). The number of A, B-frames is polynomial, because our two vectors will always correspond to extreme cycles from some two states. A and B are exponential.

The following picture shows geometrically what an A, B-frame is: a set W sitting inside of a box of size A (drawn as the letter W) is copied by shifting it in two directions. The vectors by which we are shifting are bounded by B.



It turns out that two such unions of A, B-frames are equal iff they are equal in the exponentially bounded region close to 0 (Lemma 3.4 and Lemma 3.5). Together with the fact that membership checking is in NP, we get a Π_2^{P} algorithm for checking inclusion (equivalence) of Parikh images of context-free grammars (for $d = 2$).

For $d > 2$, it may be impossible to get a polynomial number of A, B-frames (a nice counterexample in Section 7), which means that Lemma 3.5 fails (the region would be bounded double exponentially). However, we can circumvent this by splitting $\mathbb{N}^d$ into *regions* — when restricted to a single region, the number of A, B-frames will be polynomial, thus allowing us to use Lemma 3.5 successfully in each region separately, again getting a Π_2^{P} algorithm for deciding equivalence.

3 Geometry of Multisets

For a set X, the elements of $\mathbb{N}^X$ are interpreted as multisets of elements of X. A set is interpreted as a multiset, thus for $Y \subseteq X$, $Y(x) = 1$ for $x \in Y$ and $Y(x) = 0$ otherwise. For $x \in X$ and $v \in \mathbb{N}^X$, $x \in v$ denotes $v(x) > 0$. We will sometimes write x instead of $\{x\}$ (a multiset containing only a single occurence of $x \in X$). $\mathbb{N}^X$ is also treated as a subset of $\mathbb{Z}^X$, $\mathbb{Q}^X$, and $\mathbb{R}^X$. For $v \in \mathbb{Q}^X$, $|v| = \sum_{x \in X} |v(x)|$, $||v|| = \max_{x \in X} |v(x)|$. We say $u \geq v$ iff $u_x \geq v_x$ for each x.

By F_X^X we denote the set of matrices with coefficients in F and dimensions indexed with elements of X: for a matrix $M \in F_X^X$, M^i, the i-th column of the matrix, is a vector in F^X. For $M \in \mathbb{Q}_X^X$ and $v \in \mathbb{Q}^X$, Mv is a vector given by $(Mv)_j = \sum_i M_j^i v_i$, and $||M|| = \max_{x \in X} ||M^x||$.

We use the notation $[0..K]$ for the set of integers from 0 to K, $\mathbb{P}$ for the set of non-negative rationals (we don't use the more standard notation of $\mathbb{Q}^+$ to avoid double upper indexing, as in $(\mathbb{Q}^+)^X$), and $[0; K]$ for the set of rationals from 0 to K. Thus, for example, $[0..K]_X^X$ denotes matrices with coefficients in $\mathbb{N}$ bounded by K.

We can add or multiply sets of scalars, vectors, or matrices, in the usual way. For example, $U + V = \{u + v : u \in U, v \in V\}$, and $M\mathbb{N}^X$ for $M \in \mathbb{Z}_X^X$ is the set of vectors which can be obtained as a linear combination of columns of M with coefficients from $\mathbb{N}$.

Lemma 3.1 *Let M be a non-degenerate matrix in $\mathbb{Z}_\Sigma^\Sigma$, and let $v \in \mathbb{Z}^\Sigma$. Then* $(\det M)v \in M\mathbb{Z}^\Sigma$.

Proof. For $v_1, v_2 \in \mathbb{Z}^\Sigma$, we say that $v_1 \equiv v_2$ iff $v_1 - v_2 \in M\mathbb{Z}^\Sigma$. The quotient group $\mathbb{Z}^\Sigma/_\equiv$ has $\det M$ elements (intuitively, for $|\Sigma| = 2$, the number of elements is equal to the area of the parallelogram given by columns of M; this intuition also works in other dimensions). Thus, $(\det M)v \equiv 0$. □

Lemma 3.2 *Let $V \subseteq [0..K]^\Sigma$ be a linearly dependent set of vectors. Then for some $\alpha \in \mathbb{Z}^V$ we have $\sum_{v \in V} \alpha_v v = 0$, where $|\alpha| = O(K^{|\Sigma|}|\Sigma|!)$, and $\alpha_v > 0$ for some v.*

Proof. Without loss of generality we can assume that V is a minimal linearly dependent set. Thus, we get $\sum_{v \in V} \beta_v v = 0$ for some rational coefficients $\beta \in \mathbb{Q}^V$. Let u be such that $|\beta_u| \geq |\beta_v|$ for each v. Let $M \in \mathbb{Z}_\Sigma^\Sigma$ be a non-degenerate matrix whose $|V| - 1$ columns are $V - \{u\}$ (we obtain a non-degenerate matrix since V was a minimal linearly dependent set; if $|V| < |\Sigma| + 1$, we fill up the remaining columns with independent unit vectors). From Lemma 3.1 we get that $(\det M)u = Mw$ for some $w \in \mathbb{Z}^\Sigma$. Let $\alpha_u = -\det M$, $\alpha_v = w_i$ where $v = M^i$, and $\alpha_v = 0$ for remaining vectors. We have that $\sum \alpha_v v = 0$. Moreover, we have that for some $q \in \mathbb{Q}$ we have $\beta_v = q\alpha_v$ for each v (for a minimal linearly dependent set, (β_v) is unique up to a constant); thus, $|\alpha_v| < |\alpha_u| = \det M = O(K^{|\Sigma|}|\Sigma|!)$ for each i. □

Definition 3.3 *An A, B-**frame** is a set of form $W + M\mathbb{N}^\Sigma$, where $W_i \subseteq [0..A]^\Sigma$, and $M_i \in [0..B]^\Sigma_\Sigma$.*

Lemma 3.4 *Let $\Sigma = \{a, b\}$. Let A, B and C be positive integers.*

For $w \in \mathbb{P}^\Sigma$ and M in $[0..B]^\Sigma_\Sigma$, let

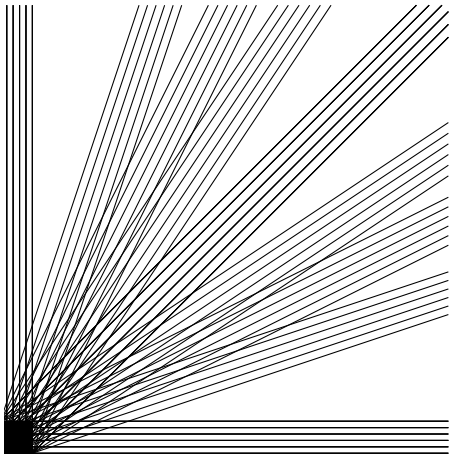
$$\mathrm{angle}(w, M) = w + M\mathbb{P}^\Sigma.$$

For $v \in \mathbb{P}^\Sigma$, the **region** *of v is defined as*

$$\mathrm{reg}(v) = \left\{ u \in \mathbb{P}^\Sigma : \begin{array}{c} \forall w \in [0; A]^\Sigma \; \forall M \in [0..B]^\Sigma_\Sigma \\ v \in \mathrm{angle}(w, M) \iff \\ u \in \mathrm{angle}(w, M) \end{array} \right\}$$

For each $v \in \mathbb{N}^\Sigma$. there exists a vector $v' \in \mathbb{N}^\Sigma$ such that $||v'|| = O(AB^2 + BC)$, $v' \in \mathrm{reg}(v)$ and $v - v' \in C\mathbb{Z}^\Sigma$.

Proof. The following picture shows this lemma graphically for $B = 3$. A is the size of the black square in the bottom left corner. Lines shown on the picture are boundaries between angles; in each bundle, 6 lines are shown, but it should be understood that there is actually a semi-line starting from each rational point in the black square.



Each angle is the set of points between two semi-lines on the picture which cross somewhere in the black square. There are three type of regions: ones containing only one vector (each bounded region is actually a singleton), 8 unbounded regions in angular shapes (between two consective bundles of lines — there are 9 bundles of lines because vectors in $[0..3]^\Sigma$ go in 9 directions), and regions in shape of semi-lines.

It can be easily calculated that each point where bundles of lines going in different directions cross has its coordinates bounded polynomially (by AB^2).

Let v be a point. If $\mathrm{reg}(v)$ is a singleton, then we are done (because v is already bounded polynomially). Otherwise, $\mathrm{reg}(v)$ is the inside of $\mathrm{angle}(w, M)$, where w is bounded polynomially and M is given by vectors in two directions M^1 and M^2 (consecutive or equal). If v is inside the paralellogram whose vertices are w, $w + CM^1$, $w + CM^2$, $w + CM^1 + CM^2$, then we are done (all those vertices are bounded polynomially). Otherwise we subtract multiplicities of CM^1 and CM^2 until we get a point w' in this paralellogram. □

Lemma 3.5 *Let $\Sigma = \{a, b\}$. For $i \in I$, let $Z_i = W_i + M_i\mathbb{N}^\Sigma$ be an A, B-frame. Let $v \in \mathbb{N}^\Sigma$. Then there exists a $v' \in \mathbb{N}^\Sigma$ such that $||v'|| = O((A + B)^{O(|I|)})$, and, for each i, $v \in Z_i$ iff $v' \in Z_i$.*

Proof. Assume that the matrices M_i are non-degenerate (the case of degenerate matrices can be solved easily by changing the matrices).

Let C be the least common multiple of determinants of matrices M_i, $C = O(B^{O(|I|)})$.

Let $v \in \mathbb{N}^\Sigma$. Let v' be the vector v' from Lemma 3.4 for our v and C; we get that $||v'|| = O((A + B)^{O(|I|)})$. We will show that it satisfies our conditions.

It is enough to check whether $v \in Z'$ iff $v' \in Z'$ for each Z' of form $w_0 + M_i\mathbb{N}^\Sigma$, where $w_0 \in W_i$.

Since M_i is non-degenerate, for some $\alpha, \alpha' \in \mathbb{Q}^\Sigma$ we have $v = w_0 + M_i\alpha$ and $v' = w_0 + M_i\alpha'$. Since v' is in the same region as v, $\alpha \geq 0$ iff $\alpha' \geq 0$. On the other hand, $v - v' \in C\mathbb{Z}^\Sigma \subseteq (\det M_i)\mathbb{Z}^\Sigma \subseteq M_i\mathbb{Z}^\Sigma$ from Lemma 3.1. Thus, $v \in \mathbb{N}^\Sigma$ iff $v' \in \mathbb{N}^\Sigma$. □

4 Commutative Grammars

Since in this paper we don't care about the order of symbols in strings generated by our grammars, we define our grammars commutatively: a state (nonterminal) produces a multiset of letters and states, not a string.

Derivation trees are defined for commutative grammars similarly as for the usual ones; we omit this definition. However, we usually also abstract from derivation trees, by considering our runs as multisets rather than trees: we don't care where in the tree each transition (production) has been used, we just count the total number of occurences. We show that there is a simple condition which checks whether our multiset corresponds to some full derivation, or a „cyclic" derivation. (A similar algebraic definition of cycles is used by the algebraic topologists.)

A **commutative grammar** is a tuple $G = (\Sigma, S, s_0, \delta)$, where Σ is a finite **alphabet**, S is a finite set of **states**, $s_0 \in S$ is an initial state, and $\delta \subseteq S \times \mathbb{N}^A \times \mathbb{N}^S$ is a set of **transitions**. We will write transitions (s, a, t) as $s \xrightarrow{a} t$; in terms of derivations, each transition consumes the state s and produces each letter from a and each state from t. For

a transition $\tau = s \xrightarrow{a} t$, $\mathrm{source}(\tau) = s$, $\mathrm{target}(\tau) = t$, and $\mathrm{out}(\tau) = a$.

We will assume that each state is a source of some transition. We will also assume that for each $s \xrightarrow{a} t \in \delta$, $|a| \leq 1$ and $|t| \leq 2$. (We do this because we want to limit things produced by the grammar in terms of $|S|$. Grammars not satisfying these conditions can be easily transformed by adding additional states.) A commutative grammar satisfying $|t| \leq 1$ is called a **regular commutative grammar** (regular grammars are equivalent to non-deterministic finite automata, with initial state s_0 and transitions with $\mathrm{target}(\tau) = 0$ as transitions to the final state; we prefer to speak about regular grammars rather than NFAs for the sake of uniformness).

For a $D \in \mathbb{N}^\delta$, $\mathrm{source}(D) \in \mathbb{N}^S$ counts how many each state appears as source of a transition: $\mathrm{source}(D)(s) = \sum_{\tau:\mathrm{source}(\tau)=s} D(\tau)$, and $\mathrm{out}(D)$ and $\mathrm{target}(D)$ counts how many each letter and each state, respectively, is produced: $\mathrm{out}(D) = \sum_\tau D(\tau)\mathrm{out}(\tau)$, $\mathrm{target}(D) = \sum_\tau D(\tau)\mathrm{target}(\tau)$. The **support** of D, $\mathrm{supp}(D) = \{s \in S : s \in \mathrm{source}(D)\}$. We say that D is **connected from** $s \in S$ if for each $t \in \mathrm{supp}(D)$ there is a path from s to t in D, i.e., a sequence $\tau_1, \ldots, \tau_m$ such that $\tau_i \in D$, $s = \mathrm{source}(\tau_1)$, $\mathrm{source}(\tau_{i+1}) \in \mathrm{target}(\tau_i)$, $t \in \mathrm{target}(\tau_m)$. We say that D is a **cycle from** $s \in S$ iff it is connected from s and it satisfies the **Euler condition**: $\mathrm{source}(D) = \mathrm{target}(D)$ (in terms of derivations, each state is consumed as many times as it is produced). We say that D is **run** iff it is connected from s_0 and $\mathrm{source}(D) = \mathrm{target}(D) + \{s_0\}$ (each state is consumed as many times as it is produced, except s_0 which is consumed one time more).

For a commutative grammar G, $\mathrm{out}(G) = \{\mathrm{out}(D) : D \text{ is a run in } G\}$.

The relation between algebraic runs and cycles and derivation trees is as follows:

Proposition 4.1 *Let G be a commutative grammar. Then:*

- *D is a run iff there is a derivation tree from s_0 where each transition τ appears $D(\tau)$ times, and all the branches are closed,*
- *D is a cycle from s iff there is a derivation tree from s where each transition τ appears $D(\tau)$ times, and all the branches are closed except one with state s at its end (we call such derivation tree* **cyclic***).*

Proof of Proposition 4.1. We show the proof for runs (for cycles the proof is similar).

Start with s_0 and try applying transitions from D (obviously, using each transition as many times as it appears in D) as long as we have some open branches. If we have used all the elements of D in the process, we are done. Otherwise, since the run D is connected, there must be some state s such that D contains some transition from s which is still not used, and s already appears in our derivation constructed so far. Since each derivation uses each state as many times as it was produced, and so does D, there also must be a yet unused transition τ_1 in D which produces s, from, say, s_1. If $s \neq s_1$, for the same reason there must be a yet unused transition τ_2 in D which produces s_1 from some s_2. Finally, we produce some s_k from s. We create a cyclic derivation tree with transitions $\tau_k, \ldots, \tau_1$ on its main branch, closing all the side branches with remaining unused transitions from D. We insert this cycle into our tree (we have produced s in some place; we cut off the part of tree from this s, insert our cycle here, and we attach the part of tree we cut off to the open branch of our cycle). Repeat until all elements of D have been used. □

Thus, if G is a commutative version of some context-free grammar H, then $\mathrm{out}(G)$ equals the Parikh image of $L(H)$, i.e., $v \in \mathrm{out}(G)$ iff there exists a $w \in L(H)$ such that each letter a appears in w $v(a)$ times.

One inclusion is obvious. In the case of regular grammars, the cycle is just what is expected (a cycle in the transition graph), and the other inclusion is equivalent to the classic theorem of Euler (characterization of graphs with Eulerian paths and cycles); in general, it is a simple generalization.

A cycle is called a **simple cycle** iff it cannot be decomposed as a sum of smaller non-zero cycles, and a run D is called a **skeleton run** if it cannot be decomposed as a sum of a run D_1 and a non-zero cycle C, where $\mathrm{supp}(D_1) = \mathrm{supp}(D)$. For each state $s \in S$, let $\mathcal{C}_s$ be the set of simple cycles from s, and $\mathcal{C}_T = \bigcup_{s \in T} \mathcal{C}_s$, for $T \subseteq S$. Also, let $\mathcal{Y}_s = \mathrm{out}(\mathcal{C}_s)$, and $\mathcal{Y}_T = \bigcup_{s \in T} \mathcal{Y}_s$ (cycle outputs).

5 Membership checking

Lemma 5.1 *Let G be a regular commutative grammar, and D be a run such that $|D| > 1 + n|\delta|$. Then $D = D_1 + nC$, where D_1 is a run, C is a simple cycle, and $\mathrm{supp}(D_1) = \mathrm{supp}(D)$.*

In case of $n = 1$, we get a limit on the size of a skeleton run.

Proof. Let τ be a transition such that $D(\tau) \geq 1 + n|\delta|$. We have $|\mathrm{target}(\tau)| = 1$ (it cannot be greater because G is regular, and cannot be 0 becuase each run in a regular grammar has exactly one transition with $|\mathrm{target}(\tau)| = 0$). Let $t \in \mathrm{target}(\tau)$. Let T be the set of states which can be reached from t via a path using only transitions τ_i such that $D(\tau_i) > n$. If $\mathrm{source}(\tau) \in T$, this finishes the proof (we have found a cycle in the graph of transitions, which can be easily translated to an algebraic cycle). Otherwise, let $u = \sum_{\tau \in D:\mathrm{source}(\tau) \notin T, \mathrm{target}(\tau) \in T} D(\tau)$, and

$v = \sum_{\tau \in D : \mathrm{source}(\tau) \notin T, \mathrm{target}(\tau) \in T} D(\tau)$. From the Euler condition, we get that $u = v$. Since τ is counted in u $1+n|\delta|$ times, and there are $|\delta|$ transitions, there must exist a transition τ' which is counted $n+1$ times in v. This is a contradiction (we have found a path from t to $\mathrm{target}(\tau') \notin T$). □

Theorem 5.2 *Let G be a regular commutative grammar, and $K \in \mathbb{N}^\Sigma$. Then $K \in \mathrm{out}(G)$ iff there exists a run D in G, $||D|| = O(|S|^{2|\Sigma|}|\Sigma|!)$, and simple cycles $C_1, \ldots, C_m$, $C_i \in \mathcal{C}_{\mathrm{supp}(D)}$, such that C_i are linearly independent and $K = \mathrm{out}(D) + \sum_i \alpha^i \mathrm{out}(C_i)$ for some $\alpha^1, \ldots, \alpha^m$.*

Proof.

Let D_0 be a run in G such that $K = \mathrm{out}(D_0)$. We decompose the run D_0 into a sum of simpler runs (on the same support) and simple cycles, until we get $D_0 = D_2 + \sum_{C \in \mathcal{C}_{\mathrm{supp}(D_0)}} \gamma_C C$, where D_2 is a skeleton. From Lemma 5.1 we get that $||D_2|| \leq |\delta|$. By taking out's, we get $K = \mathrm{out}(D_2) + \sum_{Y \in \mathcal{Y}_{\mathrm{supp}(D_0)}} \gamma_Y Y$, where $\gamma_Y \in \mathbb{N}$ for each Y.

Let $P \subseteq \mathcal{Y}_{\mathrm{supp}(D_0)}$ be the set of such cycle outputs Y that $\gamma_Y \geq L$ for $L = O(|S|^{|\Sigma|}|\Sigma|!)$. We can decompose K so that P is linearly independent. Otherwise, by Lemma 3.2, $\sum_{Y \in P} \alpha_Y Y = 0$ for some α_Y, $|\alpha_Y| \leq L$, and $\alpha_Y > 0$ for some Y. This allows us to transfer multiplicites between different cycles: if we take $\gamma'_Y = \gamma_Y - \alpha_Y$ for $Y \in P$, and $\gamma'_Y = \gamma_Y$ for $Y \notin P$, we have $\sum_Y \gamma_Y Y = \sum_C \gamma'_C Y$. We transfer multiplicites (i.e., replace γ with γ') until one of our cycles is no longer in P.

Let $D_1 = D_2 + \sum_{C \in \mathcal{C}_{\mathrm{supp}(D_1)} - P} \gamma_C C$. Since $||D_2|| \leq |\delta|$, $\gamma_C < L$, and there are at most $O(|S|^{|\Sigma|})$ distinct simple cycles in $\mathcal{Y}_{\mathrm{supp}(D_0)}$ up to equivalence of out's, we get that $||D_1|| \leq |\delta| + LO(|S|^{|\Sigma|})$. Now, $K = D_1 + \sum_{Y \in P} \gamma_Y Y$. □

Theorem 5.3 *For an alphabet Σ of fixed size, and a commutative regular grammar G over Σ, and $K \in \mathbb{N}^\Sigma$, the problem of deciding whether $K \in \mathrm{out}(G)$ is in P.*

Proof. The theorem 5.2 remains true if we define $\mathcal{C}_s$ and $\mathcal{Y}_s$ using *short cycles* instead of simple cycles — a cycle C is short iff $|C| \leq |S|$. This allows us to calculate sets $\mathcal{Y}_s$ for each state S using simple dynamic programming.

For each $T \subseteq S$ of size at most $|\Sigma|$, we calculate the set of possible $\mathrm{out}(D)$ with D satisfying the limit from Theorem 5.2 and $T \subseteq \mathrm{supp}(D)$. For each element of $\mathrm{out}(D)$ and each sequence of linearly independent elements of $\mathcal{Y}_T$, $Y_1 \ldots Y_m$, we check whether $K = \mathrm{out}(D) + \sum_i \alpha^i Y_i$ for some α^i, which can be done by solving a system of equations. □

Theorem 5.4 *For an alphabet Σ of fixed size, and two commutative regular grammars G and H over Σ, the problem of deciding whether $\mathrm{out}(G) \cap \mathrm{out}(H) = \emptyset$ is in P.*

Proof. Note that in the proof of Theorem 5.3 we have actually never used our assumption that outputs of our transitions are non-negative, e.g., Lemma 3.2 works as well for $V \subseteq [-K..K]^\Sigma$. Thus, we can check whether $\mathrm{out}(G)$ and $\mathrm{out}(H)$ are disjoint by checking whether $0 \notin \mathrm{out}(GH^{-1})$, where H^{-1} is obtained from H by negating outputs of all transitions, and GH^{-1} is a regular grammar obtained via the usual method of concatenating languages given by regular grammars G and H^{-1}. □

6 Inclusion checking

Lemma 6.1 *Let G be a commutative grammar over Σ. If D is a simple cycle or a skeleton run, then $\mathrm{out}(D) = O\left(2^{|S|^{O(1)}}\right)$.*

Proof. We start with the cycle case. We consider its cyclic derivation tree from Proposition 4.1.

If somewhere on the branch leading to s (the main branch) we had another s, we can easily split our cycle into two cycles (by splitting the derivation tree). A similar thing can be done if we had some state t in two places on the main branch.

A similar operation can be done when we find the same state twice on the side part of a branch (i.e. the part disjoint with the main branch).

Since we can use each state at most twice on each branch (once on the main part and once on the side part), this limits the size of a simple cycle to exponential in size of G.

The construction for skeletons is similar. Indeed, consider a skeleton run. If a state s appears $|S|+1$ times on a branch of a production tree, it means that there exist two consecutive appearances of s such that the part of tree between them can be cut off without removing any state from the support of this skeleton (otherwise each such state would have to be different and we would have $|S|+1$ states in total). □

Theorem 6.2 ("normal form") *Let $\Sigma = \{a, b\}$, and G be a commutative grammar over Σ. Then $\mathrm{out}(G) = \sum_{i \in I} Z_i$, where $|I| = |S|^{O(1)}$, and Z_i are A, B-frames, where $A, B = O\left(2^{|S|^{O(1)}}\right)$.*

Proof.

Let D be a run of G, and $T = \mathrm{supp}(D)$. For $Y \in \mathcal{Y}_T$, let $b(Y) = Y(b)/|Y|$; let Y^a and Y^b be the elements of $\mathcal{Y}_T$

with the smallest and largest $b(Y)$, respectively. We have $|Y^a|, |Y^b| = O\left(2^{|S|^{O(1)}}\right)$ from Lemma 6.1. There are at most $|S|^2$ possible pairs (Y^a, Y^b). Let $R(Y^a, Y^b)$ be the set of runs having particular Y^a and Y^b. We will show that $\mathrm{out}(R(Y^a, Y^b))$ is of form $W + M\mathbb{N}^\Sigma$, where the columns of M are Y^a and Y^b.

We decompose D as $D_2 + \sum_{C \in \mathcal{C}_T} \alpha_C C$, where D_2 is a skeleton. Thus, $\mathrm{out}(D)$ is decomposed as $\mathrm{out}(D_2) + \sum_{Y \in \mathcal{Y}_T} \alpha_Y Y + \beta_a Y^a + \beta_b Y^b$, where $\beta_a, \beta_b, \alpha_Y \in \mathbb{N}$. We can assume that each $\alpha_Y < \det M$, because otherwise we can replace $(\det M)Y$ by $q_a Y^a + q_b Y^b$, where $q_a, q_b \in \mathbb{N}$ (the coefficients are integers from Lemma 3.1 and non-negative since Y^a and Y^b are extreme cycles). Each Y and $\mathrm{out}(D_2)$ is $O\left(2^{|S|^{O(1)}}\right)$ from Lemma 6.1, and there are $O\left(2^{|S|^{O(1)}}\right)$ possible Y's, thus $D_3 = \mathrm{out}(D_2) + \sum_{c \in \mathrm{out}(\mathcal{C}_T)} \alpha^c c$ satisfies $||D_3|| = O\left(2^{|S|^{O(1)}}\right)$.

By taking for W the sets of possible D_3 for all runs from $R(Y^a, Y^b)$, we get the required conclusion. □

Theorem 6.3 *Let G_1 and G_2 be two commutative grammars over $\Sigma = \{a, b\}$. Then the problem of deciding $\mathrm{out}(G_1) \subseteq \mathrm{out}(G_2)$ is Π_2^{P}-complete.*

Proof. The problem is Π_2^{P}-hard because we can reduce the problem of semilinear set inclusion [Hu80] to it.

Using Theorem 6.2, we can write each $\mathrm{out}(G_k)$ as $\bigcup_{i \in I_k} Z_i^k$, where I_k is a polynomial set of indices and Z_i^k is a A, B-frame, where A and B are $O\left(2^{|S|^{O(1)}}\right)$.

From Lemma 3.5 we get that it is enough to check inclusion on vectors $v \in \mathbb{N}^\Sigma$ of size $|v| < P = O((A + B)^{O(|I_1|+|I_2|)})$. We call such vectors small vectors.

A witness for membership of v in a grammar G is a run D such that $\mathrm{out}(D) = v$, and $|v| < P$. If v is small, and D does not contain non-productive cycles (i.e., C such that $\mathrm{out}(C) = 0$; such cycles can be eliminated), then it can be described as a string of length polynomial in size of G. We call such witness a small witness.

For each v, and each small witness of membership of v in G_1, we have to find a small witness of membership of v in G_2. This can be done in Π_2^{P}. □

7 Normal form over larger alphabets?

In Theorem 6.3 we assumed that we are working with an alphabet of two letters. Does a similar statement hold for alphabets of size 3, 4, ...? What about alphabets of unfixed size?

For 2 letters, we have generated all multisets generated by our grammar from runs D having specific $\mathrm{supp}(D)$ using two extreme cycles, which led to generating $\mathrm{out}(D)$ using N^2 pairs of extreme cycles in total — Theorem 6.2. A natural conjecture is that a similar normal form exists for greater alphabets, except that there would be a polynomial (N^d) number of extreme cycles now — this would give us a straightforward generalization of Theorem 6.2, and thus also of Theorem 6.3, by combining with a generalization of Lemma 3.5. However, this is not true; in fact, Theorem 6.2 already fails for a three letter alphabet. We present this counterexample, because we think it is interesting.

Theorem 7.1 *There exists a context-free grammar G over $\{x, y, z\}$ such that $\mathrm{out}(G)$ is not a union of a polynomial number of A, B-frames.*

Proof. Consider the following grammar G (in the standard commutative grammar notation, with exponential restrictions on the size of productions):

$$\begin{array}{lcl|l} S & \to & 0 & SABCDEz \\ A & \to & B^2 & C^2D^2E^2xy \\ B & \to & C^2 & D^2E^2x^2y \\ C & \to & D^2 & E^2x^4y \\ D & \to & E^2 & x^8y \\ E & \to & 0 & x^{16}y \end{array}$$

The state S generates any number of z's together with the same number of $ABCDE$'s. $ABCDE$ generates a convex 32-gon on the surface $\mathbb{N}^{\{x,y\}}$ (we get 32 corners by deciding which transition always to use for each of five states A, B, C, D, E; they are points with coordinates $(\frac{y(y+1)}{2}, y)$ for $y \in [0..31]$). Since we generate z^n together with $(ABCDE)^n$, $\mathrm{out}(G)$ is a cone (i.e., a unbounded pyramid) with 32 edges (each edge is the line $\{(\frac{zy(y+1)}{2}, zy, z\}) : z \in \mathbb{R}\}$ for some y), and hence we need more than 16 three-dimensional A, B-frames to cover $\mathrm{out}(G)$. This example generalizes to any number of states (bigger examples are constructed using the same simple rule as the example above) — we need more than 2^{n-2} A, B-frames for a grammar with n states and two transitions for each state. Note that the n-state verison of the grammar above can be written in the limited form (i.e., for each derivation $s \xrightarrow{a} t$, $|a| \leq 1$, $|t| \leq 2$) using $O(n)$ states.

Instead of proving that this construction gives a good counterexample for each n (i.e., it indeed generates a 2^n-gon), we present another construction, based on the same idea (although we don't get as beatiful grammar as above, the proof is simpler).

For each $n \in \mathbb{N}$, we will generate a grammar G_n with n states over $\{x, y\}$, two transitions for each state, for which the set of vertices of the convex hull of $\mathrm{out}(G)$ is the set of points with coordinates $(y(N - y), y)$ for each odd $y \in$

$[0..N]$, where $N = 2^{n+1}$. It is easy to find G_1; we will now show how to construct G_{n+1} using G_n. We perform the following steps.

- We add x^{N^2} to be always generated right away from the start symbol S (i.e., to both transitions from S).
- Whenever we generate an y using some transition, we additionally generate x^{-N}, ignoring (for now) the fact that $-N$ is negative. Our vertices are now $(f(y), y)$, where $f(y) = y(N - y - N) + N^2$ for y as before. Note that $f(y) = (N - y)(N + y)$.
- We replace all occurences of y in our grammar with a new symbol Y, with two rules: $Y \to y|y^{-1}$. Our vertices are now still $(f(y), y)$, except that now y is now in range $[-N..N]$.
- We add y^N to be always generated right away from S. Now, our vertices are $(g(y), y)$, for each odd y in $[0..2N]$, and $g(y) = f(y - N) = (2N - y)y$.
- Thus, we have G_{n+1}, except that our grammar is improper due to negative transitions. However, it is easy to "normalize" our grammar: if it is possible to generate, say, x^{-a} from a non-initial state A, add x^a to the right side of each transition from A (thus eliminating x^{-a}), and replace each occurence of A on the right side of some transition with Ax^{-a}. Since the grammar is acyclic, and the initial state S never generates a negative number of any terminal, this algorithm will eventually eliminate all the negative transitions. □

8 Inclusion for fixed alphabets over more than 2 letters

The proof of the generalization Theorem 6.3 to alphabets of larger (but still fixed) size is very long and technical. We had to omit most proofs for space reasons.

In this proof, we will require lots of constants; some of them are dependant on other. To keep our constants ordered, and make sure that there is no circular reference between them, we will name them consistently $C_1, C_2, C_3, \ldots$ through the whole section; each constant will be defined in such a way that it will depend single exponentially on the size of the grammar and/or polynomially on the lower numbered constants. By induction, all numbered constants depend single exponentially on the size of the grammar. As usual, when we say *C_i is polynomial in C_j*, we assume that the size of alphabet Σ is fixed. (If $|\Sigma|$ is not fixed, then $C_i = O(C_j^{p(|\Sigma|)})$, where p is a polynomial.)

By $B^{\oplus A}$ we denote $\{\sum_{b \in B} \alpha_b b : \forall b \in B\ \alpha_b \in A\}$.

Let $J_\Sigma = \{x \in \mathbb{R}^\Sigma : x \geq 0, |x| = 1\}$.

Let $F_{C_1} \subseteq [0..C_5]^1_\Sigma$ (i.e., a set of some linear functions over $\mathbb{N}^\Sigma$ with integer coefficients up to C_5) be such that for each set of $|\Sigma| - 1$ vertices $V \subseteq [0..C_1]^\Sigma$, there exists a non-zero $f \in F_{C_1}$ such that $fV = 0$. This can be done with C_5 polynomial in C_1.

Let $L_{C_3} = \{0, C_3\}^\Sigma$ be the set of vertices of the hypercube of dimension $|\Sigma|$ and edge length C_3.

Let $\mathcal{R}(C_1, C_3)$ be the set of functions from $F_{C_1} \times L_{C_3}$ to $\{-1, 0, 1\}$.

For a $r \in \mathcal{R}$, let

$$\begin{aligned}
\mathrm{reg}(r) &= \left\{ v \in \mathbb{N}^\Sigma : \begin{array}{l} \forall f \in F_{C_1} \forall l \in L_{C_3} \\ \mathrm{sgn}(fv - fl) = r_{f,l} \end{array} \right\}, \\
\mathrm{Reg}(r) &= \left\{ v \in \mathbb{R}^\Sigma : \begin{array}{l} \forall f \in F_{C_1} \forall l \in L_{C_3} \\ \mathrm{sgn}(fv - fl) \in \{0, r_{f,l}\} \end{array} \right\}, \\
\tau(r) &= \left\{ x \in \mathbb{R}^\Sigma : \begin{array}{l} x \geq 0, x \neq 0, \\ \forall f \in F_{C_1} \forall l \in L_{C_3} \\ \mathrm{sgn}(fx) \in \{0, r_{f,l}\} \end{array} \right\}.
\end{aligned}$$

The following picture (Figure A) shows what J_Σ and $\tau(r) \cap J_\Sigma$ look like for $C_1 = 2$ and $|\Sigma| = 3$. (If $t \in \tau(r)$, then also $xt \in \tau(r)$ for $x > 0$; thus, a cross of $\tau(r)$ and J_Σ gives us information about the whole $\tau(r)$.)

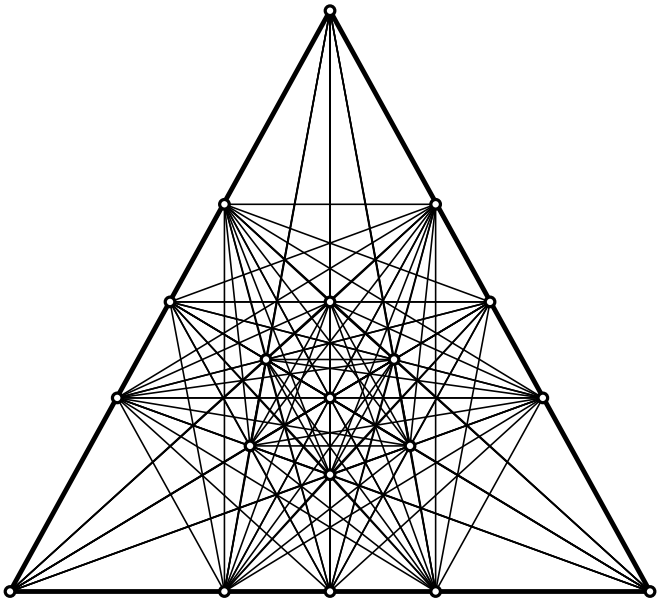


Figure A

The big equilateral triangle is J_Σ. The 19 small white circles are points $v/|v|$ for $v \in [0..C_1]^\Sigma$. We connect each pair of points with a line; these lines correspond to elements of F_{C_1}.

For each r, $\tau(r) \cap J_\Sigma$ is a part of the triangle defined by their relationship with each line (above, below, or on the line). Thus, each $\tau(r)$ is either an empty set, or one of the points where lines cross (including the 19 circles), or

a line segment between two consecutive points where the lines cross, or a polygon bounded by lines.

What does the subdivision of $\mathbb{N}^\Sigma \cap KJ_\Sigma$ into regions (reg and Reg) look like for a large K? The picture would be similar to the picture of $\tau(r) \cap J_\Sigma$, except that instead of each line we would have a bundle of lines corresponding to picking different elements of L_{C_3}. In case of $|\Sigma| = 2$ the subdivision of $\mathbb{N}^\Sigma$ into regions is similar to the picture from Lemma 3.4 (the bundles of lines are no longer infinite).

Lemma 8.1 *Let $C_1, C_3, C_4 \in \mathbb{N}$. There exist constants C_7 polynomial in C_1, and C_8 polynomial in C_1, C_3 and C_4, such that for each $r \in \mathcal{R}(C_1, C_3)$, for each $v \in \mathrm{reg}(r)$, if $||v|| \geq C_8$, then $v = v_0 + C_4 t$, where $t \in \tau(r) \cap [0..C_7]^\Sigma$ and $v_0 \in \mathrm{reg}(r)$.*

Note that a $|\Sigma|$-dimensional version of Lemma 3.4 follows easily from Lemma 8.1. Lemma 8.1 also plays a similar role in our proof as Lemma 3.4 did for $d = 2$.

Lemma 8.2 *Let $C_7 \in \mathbb{N}$. Then there exists C_9 polynomial in C_7 such that:*

Let $P, Q \subseteq [0..C_7]^\Sigma$ such that $P^{\oplus\mathbb{P}} \cap Q^{\oplus\mathbb{P}} = \{0\}$, and $0 \notin P, Q$. Then there is a $\Phi \in \mathbb{N}^1_\Sigma$ (i.e., a linear function over $\mathbb{N}^\Sigma$ with integer coefficients) such that $\Phi P > 0$, $\Phi Q < 0$, and $|\Phi| \leq C_9$.

Intuitively, this lemma states that, given two disjoint closed convex polygons in some space (in our case, the space is J_Σ, and the polygons are intersections with $P^{\oplus\mathbb{P}}$ and $Q^{\oplus\mathbb{P}}$), we can separate them strictly with a hyperplane. Such separation is a well known property of convex sets; Lemma 8.2 gives a polynomial bound on the coefficients of such a separating hyperplane.

Lemma 8.3 *Let $C_1, C_3 \in \mathbb{N}$. Then there exists a constant C_{11} such that:*

Let $S = W + \mathcal{Y}^{\oplus\mathbb{N}}$, where $W \subseteq [0..C_3]^\Sigma$ and $\mathcal{Y} \subseteq [0..C_1]^\Sigma$. Let $r \in \mathcal{R}(C_1, C_3)$. Then there exists a matrix $M \in [0..C_1]^{\Sigma}_{\Sigma}$ such that $S \cap \mathrm{reg}(r) = (W_1 + M\mathbb{N}^\Sigma) \cap \mathrm{reg}(r)$, where $W_1 \subseteq [0..C_{11}]^\Sigma$.

Theorem 8.4 *Let G be a commutative grammar, and let r be a region.*

The intersection of $\mathrm{out}(G) \cap \mathrm{reg}(r)$ is an intersection of $\mathrm{reg}(r)$ and a polynomial union of C_{11}, C_1-frames, where C_{11} and C_1 are single polynomial in $|G|$.

Theorem 8.5 *Inclusion is Π^{p}_2-complete for fixed Σ.*

Proof of Theorem 8.5. We apply the methods of Theorem 6.3 separately for each region from Theorem 8.4. □

9 Lower bounds

For completeness, we provide proofs of lower bounds for the complexities of considered problems. These results have been previously known (e.g., [Hu84]).

Theorem 9.1 *For a commutative regular grammar G over Σ (whose size is not fixed), and $K \in \mathbb{N}^\Sigma$, the problem of deciding whether $K \in \mathrm{out}(G)$ is NP-complete.*

Proof. The problem is obviously in NP (the run is the witness — the only problem is that it could be larger than polynomial by including a large number of transitions which produce nothing, but such transitions must form cycles which can be easily eliminated). We show a reduction from the Hamiltonian circuit problem. Let (V, E) be a graph. We take $\Sigma = S = V$, and for each edge (v_1, v_2) we add a transition $v_1 \xrightarrow{v_2} v_2$. We pick an initial state s_0 and add a final transition $s_0 \xrightarrow{0} 0$. The graph (V, E) has a Hamiltonian circuit iff $(1, 1, \ldots) \in \mathrm{out}(G)$. □

The same example shows that disjointness is co-NP-hard for grammars over alphabets of unfixed size. It is also co-NP-complete, since our polynomial algorithm for fixed size alphabets can be easily modified to work in co-NP for unfixed size ones.

Theorem 9.2 *For a single letter alphabet $\Sigma = \{a\}$ and a commutative grammar G (not necessarily regular), and $K \in \mathbb{N}^\Sigma$, the problem of deciding whether $K \in \mathrm{out}(G)$ is NP-complete.*

Proof. The problem is in NP for a similar reason. We can reduce the knapsack problem: given a sequence of positive integers $k_1, \ldots, k_m$ and K, is there a subset $L \subseteq \{1, \ldots, m\}$ such that $\sum_{l \in L} k_l = K$? Indeed, it is easy to produce a grammar of size $O(\sum_i \log(k_i))$ which generates Ka iff $K = \sum_{l \in L} k_l$ for some L. □

Theorem 9.3 *Let G be a commutative regular grammar over Σ of fixed size. Then the problem of deciding universality ($\mathrm{out}(G) = \mathbb{N}^\Sigma$) is coNP-hard even for $\Sigma = \{a\}$.*

Proof. The problem is in coNP because the witness for non-universality is of polynomial length (by the same argument as in Theorem 6.3).

We reduce the 3CNF-SAT problem. Let $\phi = \bigwedge_{1 \leq i \leq k} C_i$ be a 3CNF-formula with n variables $x_1 \ldots x_n$ (which can be 0 or 1) and k clauses. Let $p_1, p_2, \ldots, p_n$ be n distinct prime numbers. Let $i \in [1..k]$. Suppose that clause C_i is of form $\bigvee_{k \in [1..3]} x_{a_k} = v_{a_k}$. Our grammar will have states

S^i_j, where $0 \leq j < M_i = p_{a_1} p_{a_2} p_{a_3}$; we have cyclic transitions $S^i_j \xrightarrow{a} S^i_{(j+1) mod M_i}$, and $S^i_j \to 0$ for each j satisfying $\bigvee_{k \in [1..3]} j \bmod p_{a_k} = v_{a_k}$. We also have transitions $s_0 \to S^i_0$ for each i.

From simple number theoretic arguments we get that $x \notin \mathrm{out}(G)$ iff the formula ϕ is satisfied for $x_i = x \bmod p_i$. □

□

Corollary 9.4 *Disjointness is coNP-complete for commutative context-free grammars over Σ of fixed size, and universality, equivalence, and inclusion are coNP-complete for commutative regular grammars over Σ of fixed size.*

Proof. We get that disjointness and universality for commutative grammars are coNP-hard from Theorems 9.2 and 9.3, respectively. We get the upper bounds by applying the same methods as in Theorem 8.5 (or the easier Theorem 6.3 for alphabets of size 2). In the case of equality and inclusion for regular grammars, we get rid of one level of the polynomial hierarchy by using Theorem 5.3 to decide membership. □

10 Conclusion

We have shown tight complexity bounds for the problems of membership, inclusion (equality), and disjointness of Parikh images of regular and context-free languages over alphabets of fixed size.

What about alphabets of unbounded size? Some of the problems here remain open; we do not know whether our results and methods shed much light on these problems. For example, as far as we know, equality of Parikh images of both regular and context-free commutative languages (over alphabets of unfixed size) is only known to be Π^{P}_2-hard and in coNEXPTIME [Hu85]. Also, for the universality problem for commutative context-free grammars over alphabets of fixed size, our bounds are not tight: we know that this problem is in Π^{P}_2(as a special case of inclusion), but the only lower bound known to us is coNP (from the regular version).

In some places in our paper, it was convenient to use grammars which could produce negative quantities of letters (Theorem 5.4), or even negative quantities of states (Theorem 7.1). It is interesting whether there exists some more general theory for such techniques.

Many thanks to Sławek Lasota for introducing me to these problems, and to everyone on our Automata Scientific Excursion for the great atmosphere of research.

References

[BM99] C. Beeri, T. Milo, *Schemas for Integration and Translation of Structured and Semi-Structured Data.* ICDT 1999

[EN93] J. Esparza, M. Nielsen, *Decidability issues for Petri nets – a survey.* Bulletin of the EATCS Vol. 52 (1993), pages 245–262.

[Ha76] M.H.T. Hack. *Decidability Questions for Petri Nets.* Ph. D. Thesis, M.I.T., 1976.

[Hu80] Thiet-Dung Huynh. *The Complexity of Semilinear Sets.* ICALP 1980, LNCS 85, pages 324–337.

[Hu84] Thiet-Dung Huynh, *Deciding the inequivalence of context-free grammars with 1-letter terminal alphabet is Σ^{P}_2–complete.* Theoret. Comput. Sci. 33 (1984), pages 305–326.

[Hu85] Thiet-Dung Huynh. *Complexity of equivalence problems for commutative grammars.* Inform. and Control 66 (1985), pages 103–121.

[Hu86] Thiet-Dung Huynh. *A simple proof for the Σ^{P}_2 upper bound of the inequivalence problem for semilinear sets.* Inform. Process. Cybernet. (EIK) 22 (1986), pages 147–156.

[Hü94] H. Hüttel, *Undecidable equivalences for basic parallel processes.* Lecture Notes in Computer Science, Vol. 789. Springer, 1994. Pages 454–464.

[Ko82] S.R. Kosaraju, *Decidability of Reachability in Vector Addition Systems.* 14th ACM Symposium on Theory of Computing, San Francisco, 1982. Pages 267–281.

[Par66] Rohit J. Parikh. *On context-free languages.* Journal of the Association for Computing Machinery, 13(4):570-581, 1966.

[Pet81] J. Peterson, *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Rei85] W. Reisig, *Petri Nets: An Introduction.* EATCS Monographs in Computer Science, Vol. 4 (Springer-Verlag, Berlin, 1985)

[NS99] F. Neven, T. Schwentick, *XML schemas without order.* Unpublished, 1999.

[Sto77] L. Stockmeyer, *The polynomial-time hierarchy.* Theoret. Comput. Sci. 3 (1977), pages 1-22.

[Yen96] Hsu-Chun Yen, *On reachability equivalence for BPP-nets.* Theoret. Comput. Sci. 179 (1996), pages 301–317.

A Equality of context-free grammars for a fixed $d > 2$ — proof details

Proof of Lemma 8.1. Let $r \in \mathcal{R}$.

For all regions $\mathrm{reg}(r)$ bounded by some M (which must be polynomial), we can take $C_8 = M, C_7 = 0$.

Let v_i be a sequence of elements of $\mathrm{reg}(r)$ such that $\lim_{i\to\infty} |v_i| = \infty$. Let $w_i = v_i/|v_i|$. Let w be a cluster point of w_i. We have $w \in \tau(r)$. Thus, $\tau(r) \cap J_\Sigma$ is non-empty.

Since $\tau(r)$ is given by linear equations, we get that $\tau(r) \cap J_\Sigma$ is a polytope whose vertices (black points where edge cross in Figure A) are $t_1, \ldots, t_D$, where $t_i \in [0..C_6]^\Sigma/q_i$, where $q_i \in [1..C_6]$. Both C_6 and D are bounded polynomially.

Let R^* be the set of points $v \in \mathrm{Reg}(r)$ which cannot be written as $v_0 + t$, where $t \in \tau(r)$ and v_0 is also in $\mathrm{Reg}(r)$. It can be easily seen that $\mathrm{Reg}(r) = R^* + \tau(r)$, and also that R^* is bounded polynomially by M.

We will show that our claim is satisfied for $C_7 = {C_6}^2$ and $C_8 = M + C_4C_6D$. Let $v \in \mathrm{reg}(r), ||v|| \geq C_8$. Since $v \in \mathrm{reg}(r) \subseteq \mathrm{Reg}(r)$, we have $v = v_{00} + \sum_i \alpha_i t_i$, where $\alpha_i \geq 0$ and $v_{00} \in R^*$.

Since $||v|| > C_8$, there must be i such that $\alpha_i > C_4C_6$. Thus, $\alpha_i > C_4q_i$. We get our form: $v = v_0 + C_4q_it_i$, where v_0 is also in $\mathrm{reg}(r)$. □

Proof of Lemma 8.2. Let $||x||_2$ denote the Euclidean norm of $x \in \mathbb{R}^\Sigma$. Let $X = (J_\Sigma \cap \oplus\mathbb{P}P) - (J_\Sigma \cap \oplus\mathbb{P}Q)$; since $\oplus\mathbb{P}P$ and $\oplus\mathbb{P}Q$ are disjoint, $0 \neq X$. Let $x \in X$ be the point of x such that $||x||_2 = \min\{||x||_2 : x \in X\}$.

Now, let $x_p, x_q \in J_\Sigma$ be the points such that $x_p - x_q = x$. Let $\Phi \in \mathbb{P}^1_\Sigma$ be such that $\Phi(x_p) = 1$, $\Phi(x_q) = -1$, and $\Phi(z) = 0$ for all $z \in J_\Sigma$ such that $||z - x_p||_2 = ||z - x_q||_2$.

This Φ satisfies our conditions. We omit the proof that this construction indeed works, and that $||\Phi||$ is bounded polynomially.

Proof of Lemma 8.3.

Let C_4 be the bound on $|\det M|$ for $M \in [0..C_1]^\Sigma_\Sigma$.

Let C_7 and C_8 be from Lemma 8.1 (for our C_1, C_3, and C_4).

Let C_9 be from Lemma 8.2 (for our C_7).

Let C_{10} be such that for each $v \in [0..C_9]^\Sigma$, and each $P \subseteq [0..C_1]^\Sigma$, if $v \in P^{\oplus\mathbb{P}}$, then $Xv \in P^{\oplus[0..C_{10}]}$ for some $X \in [0..C_4]$.

Let C_{11} be big enough.

Let $r \in \mathcal{R}(C_1, C_3)$. For r such that $\mathrm{reg}(r)$ is bounded (by M_0), $M = 0$ and $C_{11} = M_0$. Thus, assume then $\mathrm{reg}(r)$ is unbounded.

Let $H = \{w/|w| : w \in [0..C_1]^\Sigma\}$. $\mathcal{Y}\mathbb{P}^\Sigma \cap J_\Sigma$ is a convex polytope with vertices from H. On the other hand, $\tau(r) \cap J_\Sigma$ is a convex polytope bounded by hyperplanes going through sets of $|\Sigma| - 1$ vertices from H; moreover, it is a minimal such polytope, i.e., it cannot be subdivided into two such polytopes of the same dimension by such a hyperplane. Thus, either $\mathrm{int}(\tau(r)) \cap J_\Sigma$ is disjoint with $\mathcal{Y}\mathbb{P}^\Sigma \cap J_\Sigma$ (case 1), or $\tau(r) \cap J_\Sigma$ is a subset of $\mathcal{Y}\mathbb{P}^\Sigma \cap J_\Sigma$ (case 2).

In the case (1), there must be a hyperplane separating $\tau(r) \cap J_\Sigma$ and $\mathcal{Y}\mathbb{P}^\Sigma \cap J_\Sigma$. Let $f \in F_{C_1}$ be such that $f(\tau(r)) \geq 0$, $f(t) > 0$ for some $t \in \tau(r)$, and $f(\mathcal{Y}) \leq 0$. Since $f(t) > 0$, we have $r(f, l) = 1$ for all $l \in L_{C_3}$. Thus, for $v \in S \cap \mathrm{reg}(r)$, we have $f(v) > f(w_0)$ for all $w_0 \in [0..C_3]^\Sigma$. On the other hand, for some w_0 we have $v = w_0 + \sum_{Y\in\mathcal{Y}} n_Y Y$, thus $f(v) \leq f(w_0)$. A contradiction. Thus, $S \cap \mathrm{reg}(r) = \emptyset$.

In the case (2), there must be a matrix M, whose columns are $|\Sigma|$ elements of $\mathcal{Y}$, such that $\tau(r) \subseteq M\mathbb{P}^\Sigma$.

Let $v \in S \cap \mathrm{reg}(r)$. We prove inductively by v.

If $||v|| \leq C_{11}$, we are ready.

Otherwise, using Lemma 8.1 iteratively, we write v as $v_0 + C_4(t_1 + \ldots + t_K)$, where $t_i \in [0..C_7]^\Sigma$, and $||v_0|| \leq C_8$. We have $K > (C_{11} - C_8)/C_4C_7$.

On the other hand, we can write v as $w_0 + \sum_{Y\in\mathcal{Y}} n_Y Y + \sum_{Y\in P} m_Y Y$, where $w_0 \in [0..C_3]^\Sigma$, $n_Y \leq C_{10}$, $m_Y \geq C_{10}$, $|P| \leq |\Sigma|$. (We get this form just like in the proof of Theorem 5.2.)

If for some i we have $t_i \in P^{\oplus\mathbb{P}}$, then we are done. Indeed, from definition of C_{10} we have that $C_4t_i = \sum_{Y\in P} \alpha_Y Y$, where $\alpha_Y < C_{10}$. On the other hand, $C_4t_i = \sum M^i\beta_i$, $\beta_i \in \mathbb{N}$. From the induction hypothesis we can present $v - C_4t_i$ in our form F. Thus we can also present v as $F + \sum_i M^i\beta_i$.

Now, what if $t_i \notin P^{\oplus\mathbb{P}}$? From Lemma 8.2, let Φ be such that $\Phi(t_i) < 0$, $\Phi(P) > 0$, $||\Phi|| < C_9$. We have:

$$\begin{aligned}
\Phi(v) &= \Phi(v_0) + C_4 \sum_i \Phi(t_i) < C_8C_9 - C_4K \\
\Phi(v) &= \Phi(w_0) + \sum_{Y\in\mathcal{Y}} n_Y Y + \sum_{Y\in P} m_Y Y > \\
&\quad -C_3C_9d - |\mathcal{Y}|C_{10}C_1C_9d + \\
&\quad +((C_{11} - C_3 + |\mathcal{Y}|C_{10}C_1)/C_1)
\end{aligned}$$

This is a contradiction for C_{11} big enough.

□

Proof of Theorem 8.4.

Let C_1 be the bound on the size of a simple cycle, i.e., $\mathcal{Y}_S \subseteq [0..C_1]^\Sigma$ (Lemma 6.1).

Let C_2 be such that for each run D we have $\mathrm{out}(D) = \mathrm{out}(D_0) + \sum_{Y\in P} n_Y Y$, where $||D_0|| \leq C_2$, and P is a subset of $\mathcal{Y}_{\mathrm{supp}D}$ of size $|\Sigma|$. (We get this form and a polynomial bound for C_2 just like in the proof of Theorem 5.2.)

Let I be the set of all subsets of S containing at most $|\Sigma|$ elements. For $i \in I$, we can create $W_i \subseteq [0..C_3]^\Sigma$ so that

$\mathrm{out}(G) = \bigcup_{i \in I} W_i + \mathcal{Y}_i^{\oplus \mathbb{N}}$. The method is similar to the one used in the proof of Theorem 6.2.

Use constants just like in Lemma 8.3.

Applying Lemma 8.3 to each component of the union, we get that for each r, $\mathrm{reg}(r) \cap \mathrm{out}(G) = \mathrm{reg}(r) \cap \bigcup_{i \in I} W_i' + M_i \mathbb{N}^{\Sigma}$. □